LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in CSE

Submitted By

Name:-APRATIM DUTTA

Roll No: 2205274

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

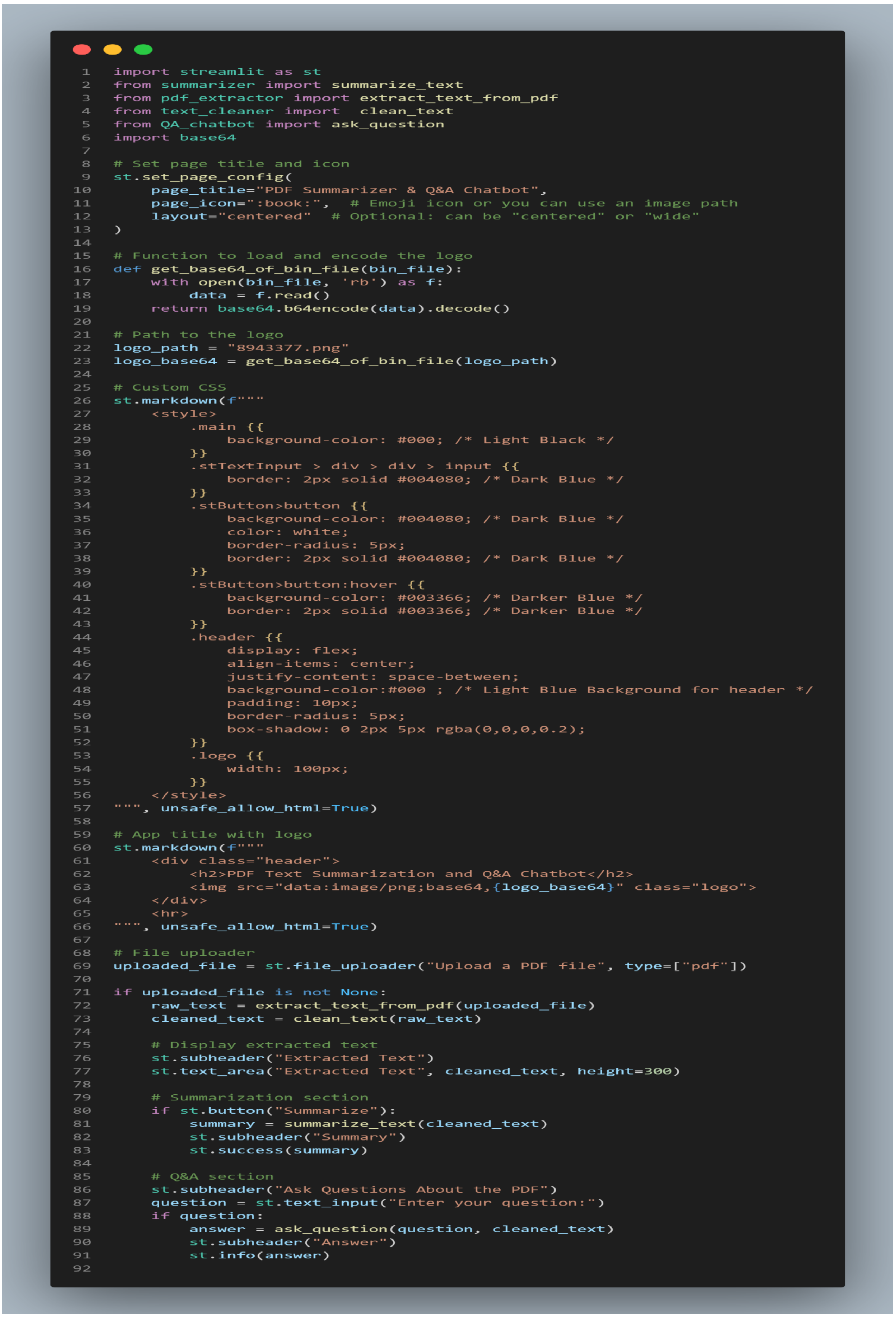
Experiment Number	5
Experiment Title	Experiment 5: Conversational Chatbot with PDF Reader
Date of Experiment	06/02/2025
Date of Submission	20/02/2025

1. Objective:- To build a chatbot capable of answering queries from an uploaded PDF document.

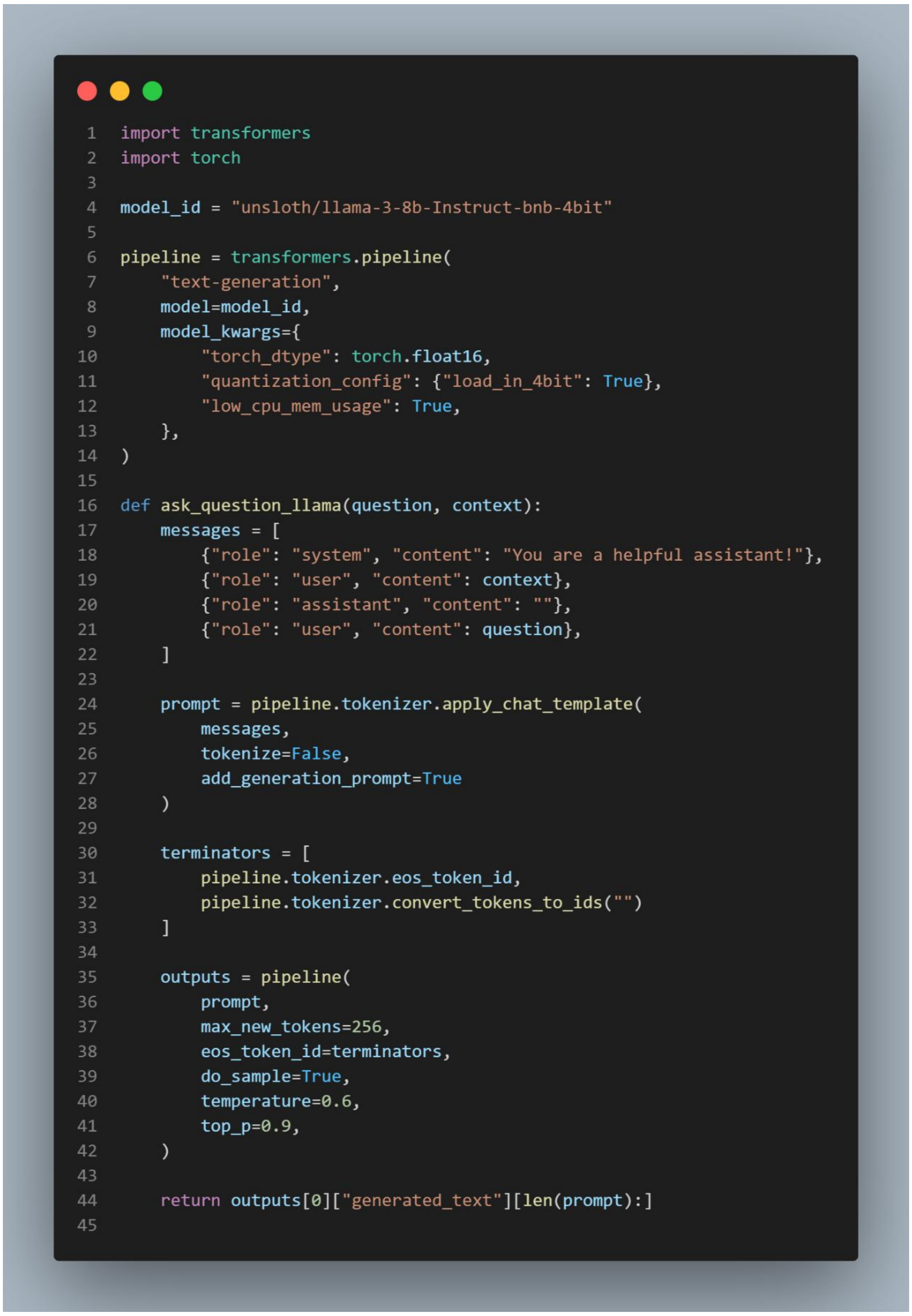
2. Procedure:-

1. Integrate open-source LLMs such as LLama or Gemma from Ollama
2. Develop a Flask backend to process the PDF/word/excel content.
3. Implement Natural Language Processing (NLP) to allow queries. You can use LLamaIndex or Langchain
4. Create a frontend to upload document files and interact with the chatbot, just like OpenAI interface
5. Provide an option to choose the LLM model from a dropdown list.
6. Display the chatbot responses on the webpage.

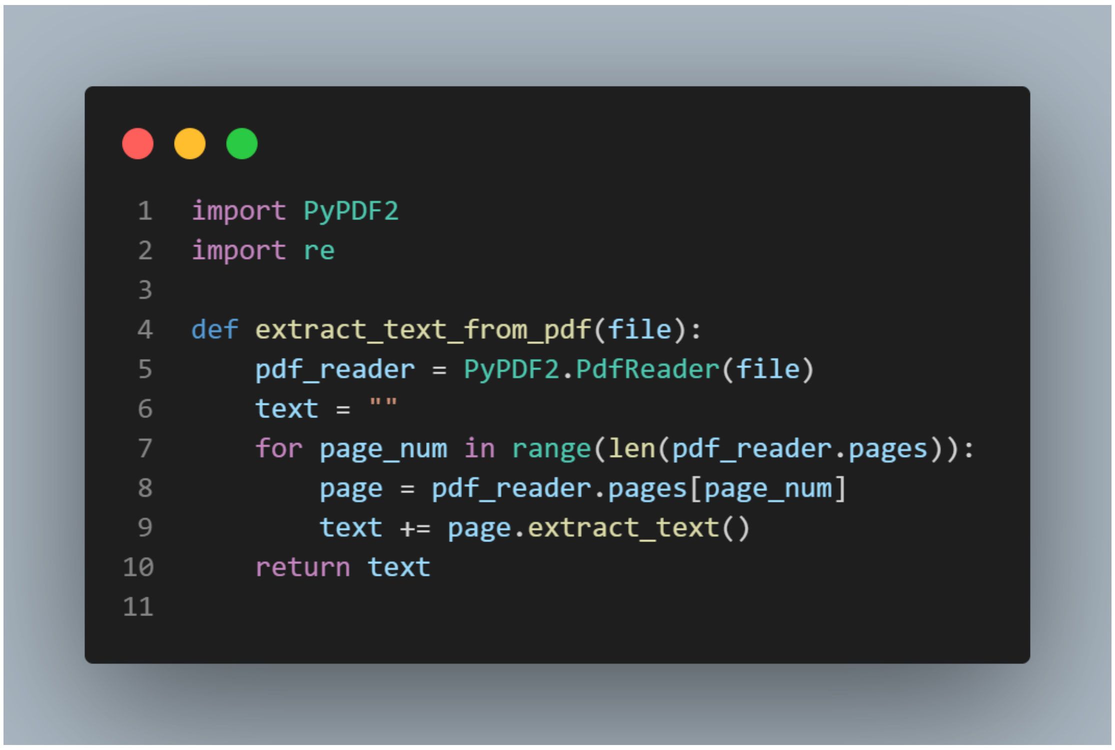
3. Code:- From next page as follows
app.py, llama_qa.py, pdf_extractor.py, QA_chatbot.py, summarizer.py, test.py, test_cleaner.py



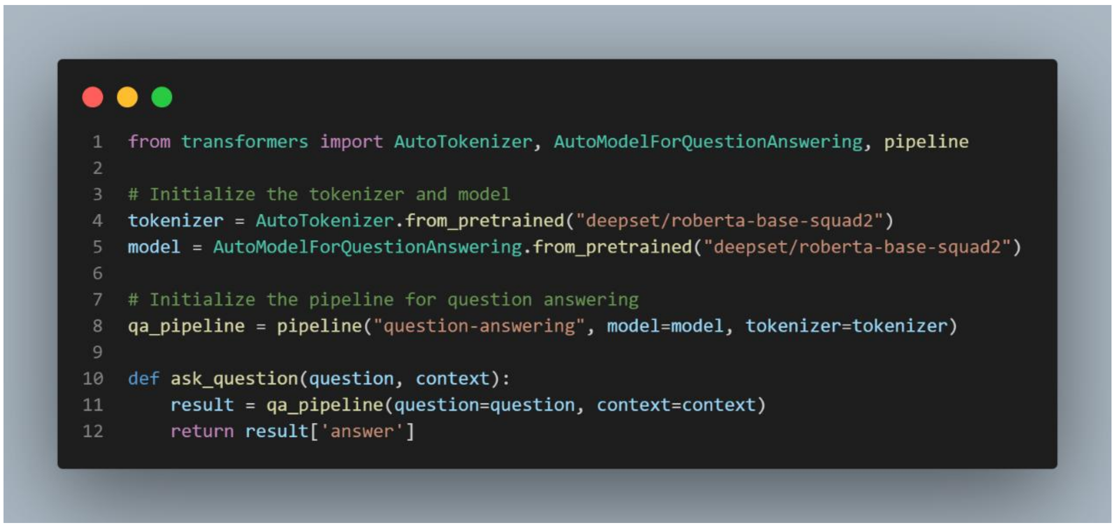
```python
import streamlit as st
from summarizer import summarize_text
from pdf_extractor import extract_text_from_pdf
from text_cleaner import  clean_text
from QA_chatbot import ask_question
import base64

# Set page title and icon
st.set_page_config(
    page_title="PDF Summarizer & Q&A Chatbot",
    page_icon=":book:",  # Emoji icon or you can use an image path
    layout="centered"  # Optional: can be "centered" or "wide"
)

# Function to load and encode the logo
def get_base64_of_bin_file(bin_file):
    with open(bin_file, 'rb') as f:
        data = f.read()
    return base64.b64encode(data).decode()

# Path to the logo
logo_path = "8943377.png"
logo_base64 = get_base64_of_bin_file(logo_path)

# Custom CSS
st.markdown(f"""
    <style>
        .main {{
            background-color: #000; /* Light Black */
        }}
        .stTextInput > div > div > input {{
            border: 2px solid #004080; /* Dark Blue */
        }}
        .stButton>button {{
            background-color: #004080; /* Dark Blue */
            color: white;
            border-radius: 5px;
            border: 2px solid #004080; /* Dark Blue */
        }}
        .stButton>button:hover {{
            background-color: #003366; /* Darker Blue */
            border: 2px solid #003366; /* Darker Blue */
        }}
        .header {{
            display: flex;
            align-items: center;
            justify-content: space-between;
            background-color:#000 ; /* Light Blue Background for header */
            padding: 10px;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0,0,0,0.2);
        }}
        .logo {{
            width: 100px;
        }}
    </style>
""", unsafe_allow_html=True)

# App title with logo
st.markdown(f"""
    <div class="header">
        <h2>PDF Text Summarization and Q&A Chatbot</h2>
        <img src="data:image/png;base64,{logo_base64}" class="logo">
    </div>
    <hr>
""", unsafe_allow_html=True)

# File uploader
uploaded_file = st.file_uploader("Upload a PDF file", type=["pdf"])

if uploaded_file is not None:
    raw_text = extract_text_from_pdf(uploaded_file)
    cleaned_text = clean_text(raw_text)

    # Display extracted text
    st.subheader("Extracted Text")
    st.text_area("Extracted Text", cleaned_text, height=300)

    # Summarization section
    if st.button("Summarize"):
        summary = summarize_text(cleaned_text)
        st.subheader("Summary")
        st.success(summary)

    # Q&A section
    st.subheader("Ask Questions About the PDF")
    question = st.text_input("Enter your question:")
    if question:
        answer = ask_question(question, cleaned_text)
        st.subheader("Answer")
        st.info(answer)

```



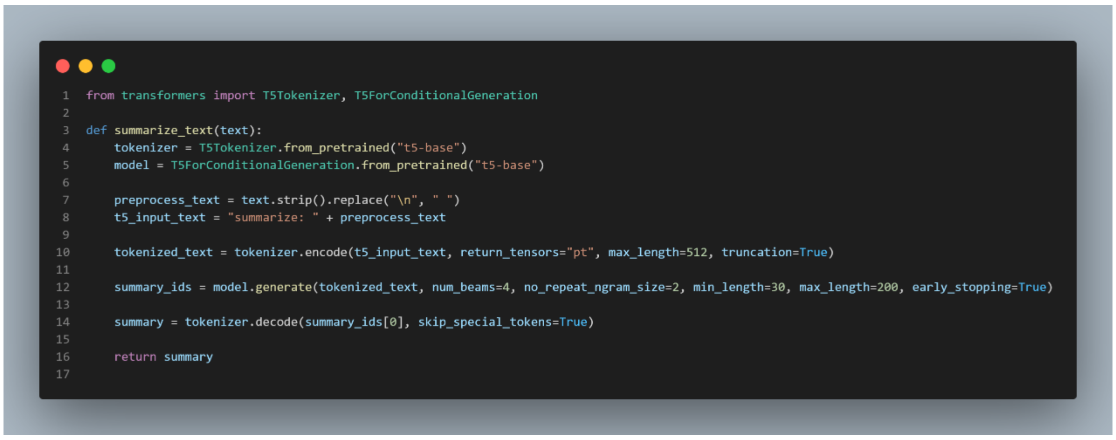
```python
import transformers
import torch

model_id = "unsloth/llama-3-8b-Instruct-bnb-4bit"

pipeline = transformers.pipeline(
    "text-generation",
    model=model_id,
    model_kwargs={
        "torch_dtype": torch.float16,
        "quantization_config": {"load_in_4bit": True},
        "low_cpu_mem_usage": True,
    },
)

def ask_question_llama(question, context):
    messages = [
        {"role": "system", "content": "You are a helpful assistant!"},
        {"role": "user", "content": context},
        {"role": "assistant", "content": ""},
        {"role": "user", "content": question},
    ]

    prompt = pipeline.tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
    )

    terminators = [
        pipeline.tokenizer.eos_token_id,
        pipeline.tokenizer.convert_tokens_to_ids("")
    ]

    outputs = pipeline(
        prompt,
        max_new_tokens=256,
        eos_token_id=terminators,
        do_sample=True,
        temperature=0.6,
        top_p=0.9,
    )

    return outputs[0]["generated_text"][len(prompt):]

```



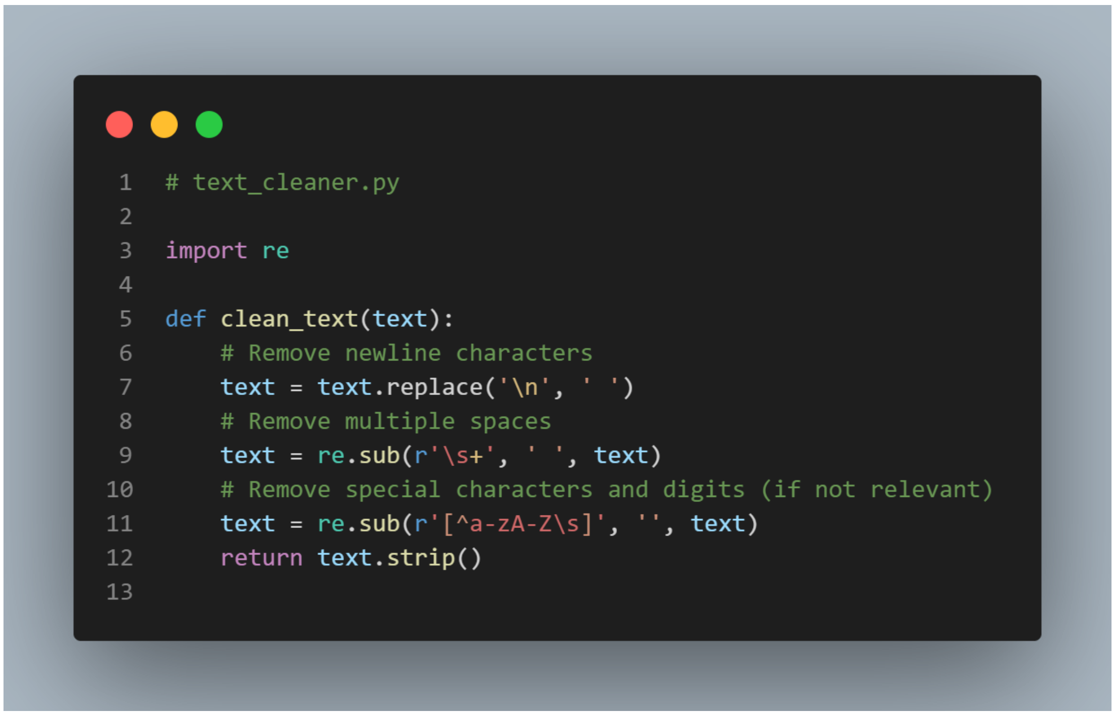
```python
import PyPDF2
import re

def extract_text_from_pdf(file):
    pdf_reader = PyPDF2.PdfReader(file)
    text = ""
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        text += page.extract_text()
    return text

```



```python
from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline

# Initialize the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("deepset/roberta-base-squad2")
model = AutoModelForQuestionAnswering.from_pretrained("deepset/roberta-base-squad2")

# Initialize the pipeline for question answering
qa_pipeline = pipeline("question-answering", model=model, tokenizer=tokenizer)

def ask_question(question, context):
    result = qa_pipeline(question=question, context=context)
    return result['answer']
```

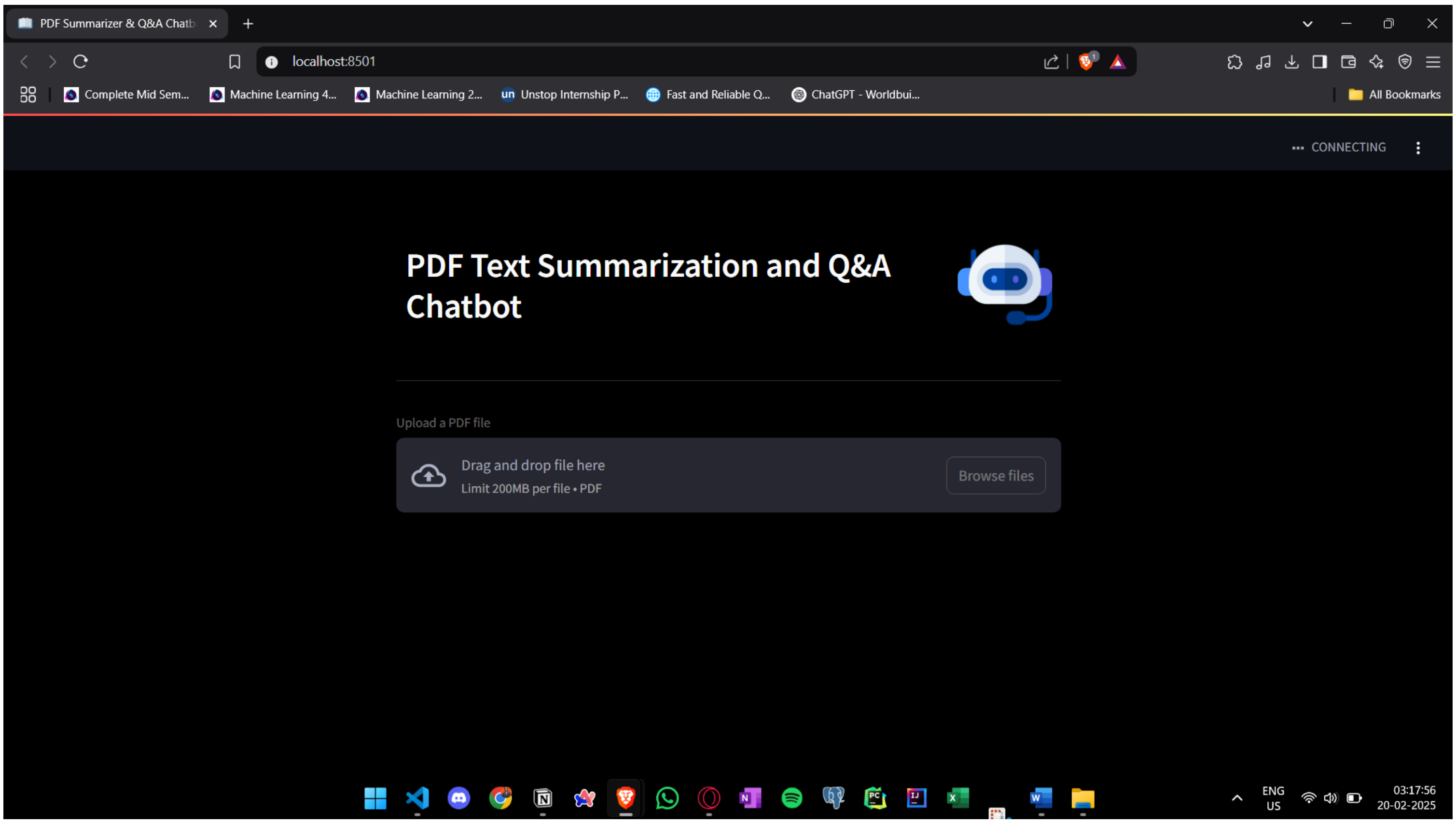


```python
from transformers import T5Tokenizer, T5ForConditionalGeneration

def summarize_text(text):
    tokenizer = T5Tokenizer.from_pretrained("t5-base")
    model = T5ForConditionalGeneration.from_pretrained("t5-base")

    preprocess_text = text.strip().replace("\n", " ")
    t5_input_text = "summarize: " + preprocess_text

    tokenized_text = tokenizer.encode(t5_input_text, return_tensors="pt", max_length=512, truncation=True)

    summary_ids = model.generate(tokenized_text, num_beams=4, no_repeat_ngram_size=2, min_length=30, max_length=200, early_stopping=True)

    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

    return summary

```



```python
# text_cleaner.py

import re

def clean_text(text):
    # Remove newline characters
    text = text.replace('\n', ' ')
    # Remove multiple spaces
    text = re.sub(r'\s+', ' ', text)
    # Remove special characters and digits (if not relevant)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text.strip()

```

4. Results/Output:- Entire Screen Shot including Date & Time



PDF Text Summarization and Q&A Chatbot

Upload a PDF file

Drag and drop file here
Limit 200MB per file • PDF

5. Remarks:-

Signature of the Student
Apratim Dutta
(Name of the Student)

Signature of the Lab Coordinator
____
(Name of the Coordinator)